tutego™
YOUR TRAINING COMPANY

Schieberegler und analoge Anzeigen

tutego™
YOUR TRAINING COMPANY

JScrollBar

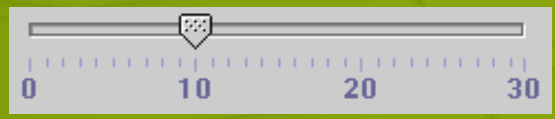
Schieberegler

- Unter Swing ersetzt javax.swing.JScrollBar die Klasse java.awt.Scrollbar.
- Direkt wird die Klasse jedoch selten verwendet.
- Man nutzt vielmehr:
  - JSlider. Schieberegler mit Beschriftung.
  - JScrollPane. Nimmt eine Komponente auf und versieht sie mit Schiebereglern.
  - JProgressBar. Fortschrittsbalken.

tutego™
YOUR TRAINING COMPANY

JSlider

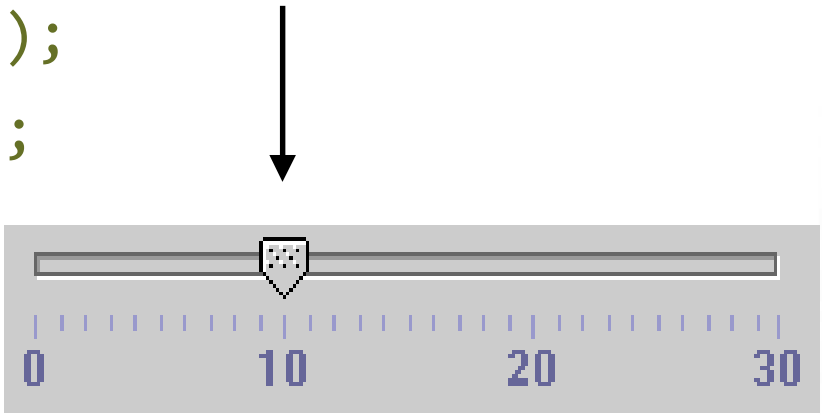
Die Klasse JSlider



- Ein JSlider lässt den Benutzer zwischen diskreten Werten mit einem Schieberegler wählen.
  - Auf diese Weise ist man unanfälliger gegen Eingabefehler.
- Die Werte sind gebunden an ein Minimum (standardmäßig 0) und Maximum (100).
- Leistungen:
  - Ein JSlider ist entweder horizontal oder vertikal angeordnet.
  - Orientierungslinien (Ticks) können gezeichnet werden.
  - Die Beschriftung lässt sich ändern.

Ein JSlider konstruieren

```
JSlider js = new JSlider( JSlider.HORIZONTAL,
                          0, 30, startwert );
js.setMajorTickSpacing( 10 );
js.setMinorTickSpacing( 1 );
js.setPaintTicks( true );
js.setPaintLabels( true );
```




```
js.addChangeListener( new SliderListener() );
```

Der ChangeListener am JSlider

- Wird der Slider bewegt, sendet er javax.swing.event.ChangeEvent-Objekte aus.
  - Ein javax.swing.event.ChangeListener fängt diese ab.
- Wird der Slider nur bewegt, aber vom Benutzer noch nicht losgelassen, liefert getValueIsAdjusting() den Wert true.

Ein SliderListener

```
class SliderListener implements ChangeListener {
    public void stateChanged( ChangeEvent e ) {
     JSlider source = (JSlider) e.getSource();
      if ( !source.getValueIsAdjusting() ) {
        // Fertig
        int result = source.getValue();
     } else {
         // Benutzer bewegt noch
     }
    }
}
```

Beschriftungen ändern

- Die Beschriftung der Werte kann man über ein Dictionary (Implementierung ist eine Hashtable) festlegen.
  - Den Werten des JSliders wird dabei ein darzustellendes Objekt zugeordnet.

```
Dictionary map = new Hashtable();
map.put( new Integer(0),  new JLabel("Stopp") );
map.put( new Integer(30), new JLabel("Langsam") );
map.put( new Integer(100),new JLabel("Schnell") );
jslider.setLabelTable( map );

jslider.setPaintLabels( true );
```

JProgressBar

Fortschrittsanzeige

- Eine lange Berechung sollte benutzerfreundlich mit einer Fortschrittsanzeige ausgestattet werden.

- Ist das Ende (oder der Fortschritt) nicht bekannt, so kann die Anzeige in ein „indeterminate mode” wechseln.

- Eine Alternative dazu sind wartende Cursor (Sanduhren).

Die Klasse JProgressBar

- Ein Konstruktor von JProgressBar nimmt das Minimum und Maximum entgehen.

```
JProgressBar progressBar;
progressBar = new JProgressBar( 0, max );
progressBar.setValue( 0 ); // aktueller Wert
progressBar.setStringPainted( true );
```

- setStringPainted() führt zur Anzeige der Prozentzahl.
- Es kann auch ein eigener String angezeigt werden.

```
if ( hälfteIstUm )
  progressBar.setString( "Halbzeit" );
```

ProgressMonitor

ProgressMonitor

- Ein JProgressMonitor ist ein Dialog, der eine Fortschrittsanzeige darstellt.


```
progressMonitor = new ProgressMonitor(
   parentComponent,          // Vater
   "Das dauert...",          // Beschreibung
   "", 0, maximum );         // Notiz, Min, Max
```

- Den Fortschritt zeigt setProgress(int) an.

Das Aufgehen des Dialogs

- Damit verhindert wird, dass bei sehr kurzen Operationen (etwa eine ½ Sekunde) Benutzer durch das Aufblinken des Dialogs verwirrt sind, geht der Dialog nicht sofort auf.
- Zwei Eigenschaften sind in diesem Zusammenhang wichtig:
  - millisToDecideToPopup (500 ms)
  - millisToPopup (2000 ms)
- Nach dem millisToDecideToPopup Millisekunden nach dem Erzeugen des Monitors vergangen sind, entscheidet er sich zur Berechung der Zeit zur Erfüllung der Aufgabe, eingerechnet dem, was schon passiert ist.
- Liegt die Zeit über millisToPopup, wird der Dialog angezeigt.

Abbruch

- Da ein ProgressMonitor ein echter Dialog ist, kann dieser abgebrochen werden.
  - Dass aber auch die Operation abgebrochen wird, ist Sache der Programmierer.
  - Ein Abbruch wird immer mit einem Dialog quittiert.
- Der Funktion isCanceled() gibt true zurück, wenn der Benutzter den Dialog abgebrochen hat.
- Nach dem Ende sollte immer die close() Methode aufgerufen werden.

Den Abbruch erkennen

```
while( !pm.isCanceled() && ! fertig ) {
  // Operation durchführen
  SwingUtilities.invokeLater( new Runnable() {
    public void run() {
      pm.setProgress( ++cnt );}
  });
}
if( pm.isCanceled() )
  JOptionPane.showMessageDialog( null,
     "Operation abgebrochen!",
     "Abbruch",
  JOptionPane.ERROR_MESSAGE );
```

ProgressMonitorInputStream

- Ein ProgressMonitorInputStream ist eine Unterklasse von java.io.FilterInputStream, der einen Fortschrittsmonitor anzeigt, wenn Daten aus einem Stream gelesen werden.

```
try {
 in = new ProgressMonitorInputStream(
              null, "Lese Datei" + fileName,
              new FileInputStream(fileName) );
}
catch( FileNotFoundException e ) {
  e.printStackTrace();
}
```

tutego™
YOUR TRAINING COMPANY

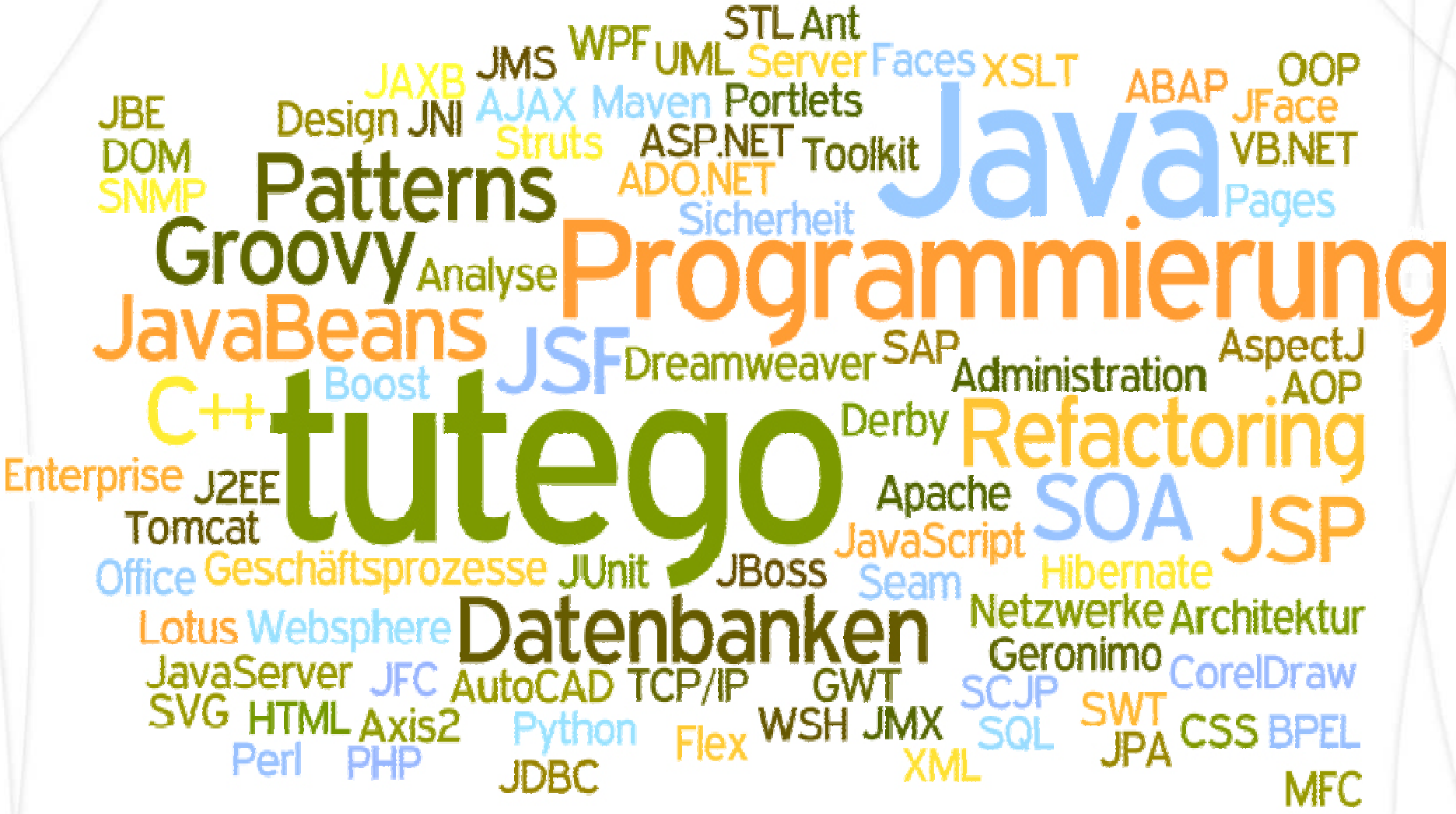
Professionelle IT-Qualifizierung

http://www.tutego.com/

tutego über tutego

- Inhouse-Schulungen mit individualisierten Inhalten und Terminauswahl
- 190 Seminare
- Kernkompetenz Java (60 Themen)
- Europaweit führende Position im Bereich Java-Schulungen
- Hochqualifizierte und zertifizierte Referenten
- Firmengründung 1997 durch Java Champion Christian Ullenboom

Unsere Themen



Unsere Themen